Unfolding Grammars in Adhesive Categories

Unfolding Grammars in Adhesive Categories*

Paolo Baldan[1], Andrea Corradini[2], Tobias Heindel[3],
Barbara König[3], and Paweł Sobociński[4]

[1] Dipartimento di Matematica Pura e Applicata, Università di Padova, Italy
[2] Dipartimento di Informatica, Università di Pisa, Italy
[3] Abteilung für Informatik und Angewandte Kognitionswissenschaft, Universität Duisburg-Essen, Germany
[4] ECS, University of Southampton, United Kingdom

Abstract. We generalize the unfolding semantics, previously developed for concrete formalisms such as Petri nets and graph grammars, to the abstract setting of (single pushout) rewriting over adhesive categories. The unfolding construction is characterized as a coreflection, i.e. the unfolding functor arises as the right adjoint to the embedding of the category of occurrence grammars into the category of grammars.
As the unfolding represents potentially infinite computations, we need to work in adhesive categories with "well-behaved" colimits of ω-chains of monomorphisms. Compared to previous work on the unfolding of Petri nets and graph grammars, our results apply to a wider class of systems, which is due to the use of a refined notion of grammar morphism.

1 Introduction

When modelling systems one often needs a truly concurrent semantics providing explicit information concerning causality, conflict and concurrency of events in computations. This is clearly the case if one wants to understand and investigate the inherent concurrency of a given system, but truly concurrent models are also a cornerstone of verification techniques based on partial order methods [17]. In fact, the latter avoid the enumeration of all possible interleavings of events, and, in this way – especially in the case of highly concurrent systems – yield very compact descriptions of the behaviour of a system.

One such partial order method is the unfolding approach: it "unravels" a system and produces a structure which fully describes its concurrent behaviour, including all reachable states and the mutual dependencies of all possible steps.

Unfolding techniques were first introduced for Petri nets and later extended to several other formalisms, e.g. to graph transformation systems, which in turn generalize (various extensions of) Petri nets. However, there are many types of graph transformation formalisms – based on undirected and directed graphs, hypergraphs, graphs with scopes, graphs with second-order edges, and so forth. Hence a more abstract notion of unfoldings is called for, which exhibits the "essence" of the unfolding technique underlying all these special cases.

* Supported by DFG project SANDS and project AVIAMO of the University of Padova.

To this aim, we propose an *abstract* unfolding procedure which applies uniformly to all these rewriting mechanisms. Following the line of research of [9, 14], we shall regard system states as objects of a category $\mathbb{C}$ satisfying suitable properties. Part of the properties of $\mathbb{C}$ ensure a meaningful notion of $\mathbb{C}$-object rewriting, while other additional properties are required to guarantee, first, that the unfolding procedure is feasible and, second, that the unfolding can be characterized as a co-reflection in the style of [20].

The approach to rewriting that we will use is the *single pushout approach* (SPO) [16]. This is one of the most commonly used *algebraic approaches* to rewriting, alternative to the *double pushout* (DPO) approach [10], where some subtle complications due to the inhibiting effects of DPO rewriting are avoided.

As a categorical framework we consider *adhesive categories* [14], which turn out to be appropriate for SPO rewriting as the needed pushouts in the partial map category Par($\mathbb{C}$) can be shown to exist. After having provided an algorithm to construct all finite prefixes of the unfolding, a crucial step consists in joining these parts into a single structure. To ensure that this is possible, we need that colimits of ω-chains of monomorphisms exist and satisfy suitable properties. Adhesive categories having sufficiently well-behaved colimits of ω-chains of monomorphisms will be called ω-*adhesive* (see also [12, 8]).

The main result states that the unfolding construction induces a *coreflection*, i.e. it can be expressed as a functor that is right adjoint to the embedding of the category of occurrence grammars, the category where unfoldings live, into the category of all grammars. In order to define the category of grammars we introduce an original notion of grammar morphism which is similar to the graph grammar morphisms proposed in [4] but more concrete; as a consequence, we can treat uniformly the whole class of grammars, without the need to restrict to so-called *semi-weighted* grammars as it was done in several approaches for Peri nets (see, e.g. [18]) and for graph grammars [4].

Roadmap: In order to motivate at a more intuitive level the definitions and constructions which will follow, we first sketch the general ideas of our work. Note that we work in a setting of abstract objects (which could be sets, multisets, graphs, etc.) which are rewritten according to a rule by removing (the image of) its left-hand side and by gluing its right-hand side to the remaining object. According to the SPO approach, the left- and right-hand sides of a rule are related by a partial map, i.e. a span $L \hookleftarrow K \rightarrow R$ in the underlying category where the left leg is a mono. As it is usually done in unfolding approaches we restrict to linear rules where both legs are mono; for a schematic representation see Figure 1(a).

A rule essentially indicates what is deleted (▤), what is preserved (■) and what is created (▥). This can either be represented by a span as in Figure 1(a) or by a combined representation (see Figure 1(b)). Very roughly, given a (linear) rule $L \hookleftarrow K \rightarrowtail R$ (or $L \supseteq K \subseteq R$ in a more set-based notation), an object G that contains the left-hand side L is rewritten to $G \backslash (L \backslash K) \cup R$. This however is properly defined only if the complement exists.

In a general setting the so-called dangling condition issue arises. In the case of graphs, the dangling condition can be understood as follows: what happens if

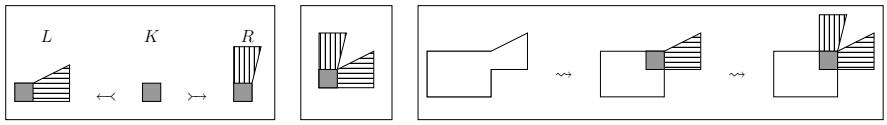


Fig. 1. (a) Rule as a partial map; (b) Combined rule representation; (c) Schematic representation of an unfolding step

a node is to be removed, and such a node is attached to an edge which is not explicitly deleted? There are two ways to resolve this issue: the DPO solution which forbids the rewriting step, and the SPO solution which removes the edge. Since the inhibiting effects of the first solution lead to serious complications in the theory, we follow the latter path, which, as we will discuss, amounts to defining the term $G\backslash(L\backslash K)$ using the more general construction of relative pseudo-complements, known from lattice theory.

The unfolding of a fixed start object provides a partial order representation of the set of derivations in the grammar starting from such an object. Intuitively the construction works as follows: look for an occurrence of a left-hand side of a rule and, instead of replacing it, attach the right-hand side as in Figure 1(c) and record the occurrence of the rule. Doing this iteratively one obtains a growing object, called a type object, which is possibly infinite, and a set of rules that describe the dependencies on this type object.

Now, in order to characterize the unfolding construction abstractly and to show its universality, we will need the following concepts:

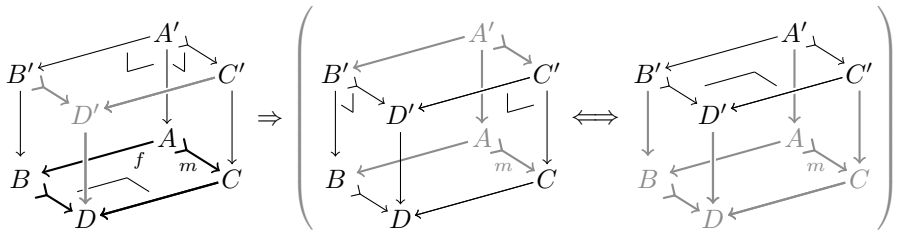
- *A notion of category which allows to define* (SPO) *rewriting properly:* For this we will use adhesive categories which can be used for defining abstractly a notion of rewriting which enjoys suitable Church-Rosser properties. (Section 2)
- *An analogue of occurrence nets:* As the unfolding of a Petri net is a special kind of net, the unfolding construction in this abstract setting will produce a special kind of grammar, which will be characterized as an occurrence grammar. In occurrence grammars suitable notions of causality, concurrency and conflict can be defined, allowing for a "static" characterization of reachable states as concurrent objects. (Section 4)
- *Well behaved ω-colimits:* In order to be able to construct potentially infinite unfoldings, we have to be able to glue together a countable chain of finite prefixes of the unfolding. To this aim we require that colimits of ω-chains exist and are well-behaved: adhesive categories enjoying this property are called ω-adhesive. The notion of ω-adhesivity is a natural extension of adhesivity that enjoys several closure properties. (Section 5)
- *A category of grammars and the coreflection result:* Finally we will present a coreflection result, i.e. we will show that the unfolding is in a sense the "best" approximation of the original grammar in the realm of occurrence grammars. In order to do this we have to introduce a category of grammars, defining a suitable notion of grammar morphism. (Section 5)

2 Adhesive categories for SPO rewriting

We will use adhesive categories [14] as a basis for the rewriting framework. For this we fix an (adhesive) category $\mathbb{C}$ to which all objects and morphisms belong.

Definition 1 (Adhesive category). *A category is* adhesive *if*

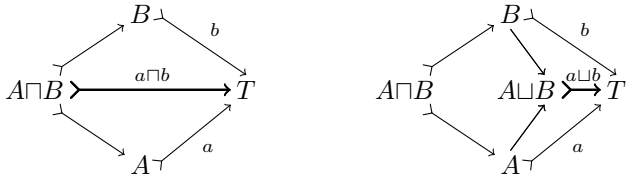
1. *it has all pullbacks;*
2. *pushouts along monomorphisms exist, i.e. for each span $B \leftarrow f - A \rightarrowtail m \rightarrow C$ with monic m, a pushout $B - n \rightarrow D \leftarrow g - C$ exists, yielding a* pushout square $\begin{smallmatrix} B & \leftarrow & A \\ D & \leftarrow & C \end{smallmatrix}$*;*
3. *all pushouts along monos are Van Kampen squares, i.e. given a cube diagram as shown below with: (i) m monic, (ii) the bottom square a pushout and (iii) the back squares pullbacks, we have that the top square is a pushout iff the front squares are pullbacks.*



It is known that every topos is adhesive [15]. The subobjects of an object in an adhesive category form a distributive lattice, a fact which we will make more precise in the following.

Definition 2 (Subobject poset). *Let $T \in \mathbb{C}$ be an object. Two monomorphisms $a\colon A \rightarrowtail T$, $a'\colon A' \rightarrowtail T$ are isomorphic if there exists an isomorphism $i\colon A \to A'$ with $a = a' \circ i$. Such an equivalence class is called the* subobject represented by a*. Then the* subobject poset $\langle \mathrm{Sub}(T), \sqsubseteq \rangle$ *has isomorphism classes $[a\colon A \rightarrowtail T]$ of monomorphisms over T as elements. Further, given two monomorphisms $a\colon A \rightarrowtail T$ and $b\colon B \rightarrowtail T$, the inclusion $[a] \sqsubseteq [b]$ holds if there exists $j\colon A \rightarrowtail B$ such that $a = b \circ j$.*

Proposition 3 (Distributive subobject lattices [14]). *Any subobject poset in an adhesive category is a distributive lattice, where the meet $[a] \sqcap [b]$ of two subobjects $[a]$, $[b]$ is obtained by taking the pullback of their representatives and the join $[a] \sqcup [b]$ is obtained by taking a pullback, followed by a pushout (i.e. adhesive categories have effective unions).*



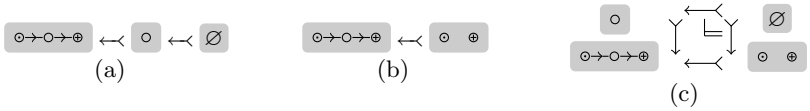
Another operation on subobjects that is directly connected to the SPO rewriting mechanism is *relative pseudo-complementation* [7] (cf. Proposition 6).

Definition 4 (Relative pseudo-complement). *Let $\langle L, \sqsubseteq\rangle$ be a lattice. The* relative pseudo-complement (RPC) of a with respect to b, *written $a \mapsto b$, is an element d satisfying $a \sqcap x \sqsubseteq b \iff x \sqsubseteq d$ for all $x \in L$. It is unique if it exists.*

The lattice L is relatively pseudo-complemented (RPC) *if the* RPC $a \mapsto b$ *exists for all pairs $a, b \in L$.*

In a finite distributive lattice the RPC of a w.r.t. b always exists and can be obtained as $a \mapsto b = \bigsqcup\{y \mid a \sqcap y \sqsubseteq b\}$. We consider the following two special cases:

- In the case of a powerset lattice, given two sets $B, A \in \wp(M)$, the RPC of A w.r.t. B, is the set $M \setminus (A \setminus B) = \{m \in M \mid m \notin A \text{ or } m \in B\}$.
- In the case of subobject lattices, if $[a], [b] \in \mathrm{Sub}(T)$, with $[a] \sqsupseteq [b]$, the RPC $[c] = [a] \mapsto [b]$ with $c\colon (A \mapsto B) \rightarrowtail T$ gives rise to a particular pullback square $\begin{smallmatrix} A & \hookleftarrow & B \\ \downarrow & \llcorner & \downarrow \\ T & \hookleftarrow & A \mapsto B \end{smallmatrix}$.
  In particular, note that that the RPC $[a] \mapsto [a]$ is $[\mathrm{id}_T]$ (if the top arrow of the pullback is an iso so is the bottom arrow).
  As an example we consider the category of directed graphs and graph morphisms (i.e., the functor category $\mathbf{Set}^{\bullet \rightrightarrows \bullet}$) which is known to be a topos and thus adhesive. Given the two graph inclusions shown in (a) below, the RPC is given in (b) and yields the pullback square shown in (c). This construction corresponds to taking the *largest pullback complement*.



(a) (b) (c)

Having relatively pseudo-complemented subobject lattices will be important for SPO rewriting. In SPO rewriting a rule is essentially a partial map which specifies what is deleted, what is preserved and what is created. Given a category $\mathbb{C}$ with pullbacks along monomorphisms, its category of partial maps is defined as follows (see [19]).

Definition 5 (Partial maps). *The* category $\mathrm{Par}(\mathbb{C})$ of partial maps (in $\mathbb{C}$) *has the same objects as $\mathbb{C}$, i.e.* $\mathrm{ob}(\mathrm{Par}(\mathbb{C})) = \mathrm{ob}(\mathbb{C})$. *An arrow in* $\mathrm{Par}(\mathbb{C})$ *is a $\mathbb{C}$-span $A \leftarrow m \prec X \xrightarrow{f} B$ with monic m, taken up to isomorphisms at X (Fig. 2(a)). It is called a* partial map *and is written $(m_{(X)} f\rangle : A \rightharpoonup B$ or just $(m, f\rangle : A \rightharpoonup B$.*

If m is an isomorphism, then $(m, f\rangle$ is called a total map. *The identity on an object A is $(\mathrm{id}_A, \mathrm{id}_A\rangle : A \rightharpoonup A$ (Fig. 2(b)); composition is defined via pullbacks (Fig. 2(c)).*

Note that a partial map $(m, f\rangle : A \rightharpoonup B$ is monic in $\mathrm{Par}(\mathbb{C})$ *if and only if it is total and f is monic in $\mathbb{C}$; hence we often write $C \rightarrowtail D$ instead of $C \xrightarrow{(\mathrm{id}, g\rangle} D$.*

Proposition 6 (Partial map pushouts). *Let $\mathbb{C}$ be an adhesive category. Then in* $\mathrm{Par}(\mathbb{C})$ *pushouts along monomorphisms exist if and only if subobject lattices in $\mathbb{C}$ are relatively pseudo-complemented.*

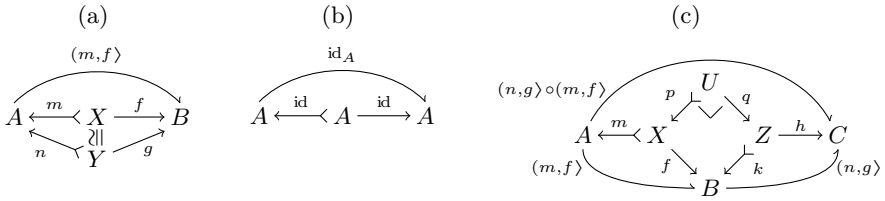


Fig. 2. Diagrams showing partial maps.

The pushout can be obtained as depicted below. Note that the vertical partial maps are monic and hence we omit the first legs which are isos. Starting from the diagram on the left, one first obtains D as RPC *of m w.r.t. $m \circ \alpha$ and then constructs C as pushout of d and β in $\mathbb{C}$.*

Besides assuming that $\mathbb{C}$ is adhesive we also require that the operation of taking RPCs is *stable under pullback* (in the sense of Lemma 1.4.13 in [13]), a fact needed later to show that SPO rewriting is preserved by grammar morphisms. All toposes and all adhesive categories known to the authors satisfy this requirement. Note that it would also be possible to develop the theory in another setting such as Heyting categories [13].

3 Grammars and grammar morphisms as simulations

The basic entities of single-pushout rewriting are rules, which are partial maps, and grammars, which are collections of rules with a start object. As it is standard in unfolding approaches, we restrict to linear and consuming rules. Further, rewriting amounts to taking pushouts along the rules of a grammar.

Definition 7 (Rules and SPO rewriting). *A* $\mathbb{C}$-rule *is a partial map span $q = L \leftarrow\!\alpha\!\prec K \prec\!\beta\!\rightarrow R$. It is called* linear *if β is monic and* consuming *if α is not an isomorphism. We denote by $\mathscr{R}_{\mathbb{C}}$ the class of consuming, linear $\mathbb{C}$-rules.*

Let $A \in \mathbb{C}$ be an object. A (monic) match *for a rule $q = L \leftarrow\!\alpha\!\prec K \succ\!\beta\!\rightarrow R$ into A is a monomorphism $m\colon L \rightarrowtail A$ in* $\mathrm{Par}(\mathbb{C})$*. Then q* rewrites *A* (at *m*) to *B, written $A \models_{\langle q,m\rangle} B$ or simply $A \models_q B$, if there is a pushout square ${}^{L}_{A}\!\downarrow\!\overrightarrow{\ulcorner}\!\downarrow^{R}_{B}$ in* $\mathrm{Par}(\mathbb{C})$*, i.e. if a pushout $A \prec\!b\!\rightarrowtail B \leftarrowtail\!n\!\prec R$ of $A \leftarrow\!m\!\prec L \prec\!(\alpha,\beta)\!\rightarrowtail R$ exists; in this situation $A \models_{\langle q,m\rangle} B$ is also referred to as an* SPO rewriting step.

Example 8. The graph $S = \circlearrowright_{\rightarrow} \circ \circlearrowright_{\rightarrow} \circ$ models a tiny network: vertices are network nodes, edges are directed network links, and looping edges represent stored

messages. Further $q_1 = \;\circlearrowright\!\!\rightarrow\!\circ \leftarrowtail \circ\!\rightarrow\!\circ \rightarrowtail \circ\!\rightarrow\!\circlearrowright\;$ and $q_2 = \circ \leftarrowtail \varnothing \rightarrowtail \varnothing$ model message dispatching and failure of network nodes, respectively. Now q_1 can rewrite S, namely $\circlearrowright\!\!\rightarrow\!\circ\!\rightarrow\!\circ \vDash_{q_1} \Rightarrow \circ\!\rightarrow\!\circlearrowright\!\!\rightarrow\!\circ$. In the latter state, the failure of the middle network node is captured by $\circ\!\rightarrow\!\circlearrowright\!\!\rightarrow\!\circ \vDash_{q_2} \Rightarrow \circ \quad \circ$, i.e. dangling edges are removed.

Definition 9 (Slice category). *For an object $T \in \mathbb{C}$, the* slice category over T*, denoted $\mathbb{C}{\downarrow}T$, has $\mathbb{C}$-morphisms $A \!-\!a\!\rightarrow\! T$ with codomain T as objects. A $\mathbb{C}{\downarrow}T$-morphism $\psi\colon (A \!-\!a\!\rightarrow\! T) \to (B \!-\!b\!\rightarrow\! T)$ is a $\mathbb{C}$-morphism $\psi\colon A \to B$ satisfying $a = b \circ \psi$. Further, we denote by $|\_|_T\colon \mathbb{C}{\downarrow}T \to \mathbb{C}$ the obvious forgetful functor, mapping $A \!-\!a\!\rightarrow\! T$ to A, and acting as the identity on morphisms.*

A grammar will be defined as a set of rules with a start object. This is in analogy to Petri nets where we regard the latter as a set of transitions with an initial marking. More precisely, as described in detail in [4], the token game of a Petri net with place set P can be modelled by SPO rewriting in the slice category $\mathbb{S}{\downarrow}P$, where $\mathbb{S}$ is the category of (finite) sets and functions: multisets are encoded as functions with co-domain P. Abstracting away from sets, we work in the slice category $\mathbb{C}{\downarrow}T$ for a given "place" object T, also called *type object*.

Example 10. Fixing the *type graph* $T = \circlearrowleft\!\!\circ\!\!\circlearrowright$, we give a typed version of Example 8. The double fins correspond to network links, i.e. the typed version of $\circlearrowright\!\!\rightarrow\!\circ\!\rightarrow\!\circ$ is $\circlearrowright\!\!\twoheadrightarrow\!\circ\!\twoheadrightarrow\!\circ$ where the morphisms into T is the unique one preserving the fins; the typed rules are given by $\circlearrowright\!\!\twoheadrightarrow\!\circ \leftarrowtail \circ\!\twoheadrightarrow\!\circ \rightarrowtail \circ\!\twoheadrightarrow\!\circlearrowright$ and $\circ \leftarrowtail \varnothing \rightarrowtail \varnothing$.

Notation: We introduce a convention for rules $q \in \mathscr{R}_{\mathbb{C}\downarrow T}$: we will always assume $q = l_q \leftarrow\!\alpha_q\!\prec k_q \succ\!\beta_q\!\rightarrow r_q$ and $|q|_T = L_q \leftarrow\!\alpha_q\!\prec K_q \succ\!\beta_q\!\rightarrow R_q \in \mathscr{R}_{\mathbb{C}}$, where the latter is the obvious untyped version of q.

If $\mathbb{C}$ is adhesive and RPCs are stable under pullback, then $\mathbb{C}{\downarrow}T$ has the same properties (cf. Proposition 29 and [14]). Now *typed* grammars are defined as follows.

Definition 11 (Typed grammar). *Let $T \in \mathbb{C}$ be an object called the* type object*. A T-*typed grammar *G is a pair $G = \langle Q, s\colon S \to T\rangle$ where $Q \subseteq \mathscr{R}_{\mathbb{C}\downarrow T}$ is a set of linear, consuming $\mathbb{C}{\downarrow}T$-rules, and $S \!-\!s\!\rightarrow\! T \in \mathbb{C}{\downarrow}T$ is the* start object*.*

The rewriting relation *over $\mathbb{C}{\downarrow}T$-objects associated with G is defined by $a \vDash_G\!\Rightarrow b$ if $a \vDash_q\!\Rightarrow b$ for some $q \in Q$; further an object $a \in \mathbb{C}{\downarrow}T$ is* reachable *in G if $s \vDash_G\!\Rightarrow^* a$, where $\vDash_G\!\Rightarrow^*$ is the transitive-reflexive closure of $\vDash_G\!\Rightarrow$.*

In the rest of the paper we restrict ourselves to *finite* grammars.

Definition 12 (Finite grammar). *Let $G = \langle Q, s\colon S \to T\rangle$ be a grammar; then G is* finite *if the start object and all left and right hand sides are finite, i.e.* $\mathrm{Sub}(S)$ *is finite and* $\mathrm{Sub}(L_q)$, $\mathrm{Sub}(R_q)$ *are finite for all $q \in Q$. Moreover for each rule there are at most finitely many other rules with isomorphic left-hand sides, i.e. the set $\{q' \mid l_q \cong l_{q'}\ \&\ q' \in Q\}$ is finite for each $q \in Q$.*

Finiteness of a grammar ensures that every reachable object is finite. As a consequence, using Proposition 6 and existence of RPCs in finite lattices, it also guarantees that for every rule q and match m of q into a reachable object, the pushout of q and m exists in $\mathrm{Par}(\mathbb{C})$: this implies that (as it is usual for the SPO approach) rewriting is possible at any match.

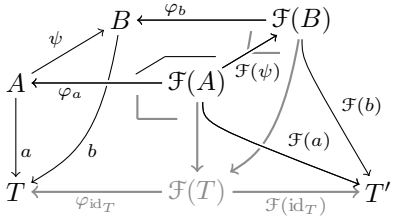
Retyping operations and grammar morphisms. Now we equip grammars with a notion of morphism, turning them into a category. Following the ideas in the literature on Petri nets and graph transformation, a morphism relating two systems should induce a simulation among them, in the sense that every computation in the source system is mapped to a computation of the target system. Another desirable property is that a notion of morphism defined in our abstract setting should "specialize" to the corresponding notions proposed for systems such as Petri nets and graph grammars. The morphisms we introduce below will satisfy the first requirement, i.e. a grammar morphism will describe how the target grammar can simulate the source grammar. Concerning the second property, the proposed notion of morphism is more concrete: for example, when $\mathbb{C}$ is the category of graphs, a graph grammar morphism of [4] might be induced by several different ones according to our definition. However, this greater explicitness allows to characterize the unfolding as a coreflection without restricting to the so-called *semi-weighted* grammars (cf. [18, 4, 1]).

In analogy to Petri nets, where morphisms are monoid homomorphisms preserving the net structure, a morphism between two grammars typed over T and T', respectively, will be a functor $\mathcal{F}\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ that preserves the rules and the start object, and comes equipped with some additional information.

Definition 13 (Retyping operation). *A* retyping operation $\underline{\mathcal{F}}\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ *is a pair* $\underline{\mathcal{F}} = \langle \mathcal{F}, \varphi \rangle$ *where* $\mathcal{F}\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ *is a functor mapping each object* $A \text{–}a\text{→} T$ *to* $\mathcal{F}(a) = \mathcal{F}(A) \text{–}\mathcal{F}(a)\text{→} T'$*, and* $\varphi\colon (|\_|_{T'} \circ \mathcal{F}) \dot{\to} |\_|_T$ *is a* cartesian natural transformation.[5]

Every morphism $f\colon T \to T'$ *in* $\mathbb{C}$ *induces a* (canonical) retyping operation $\sharp f = \langle f \circ \_, \mathrm{id}_\_ \rangle$ *where the functor* $f \circ \_\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ *post-composes any* $\mathbb{C}{\downarrow}T$*-object with* f*, and* $\mathrm{id}_\_$ *is the family of identities* $\{\mathrm{id}_A\colon A \to A\}_{(A \xrightarrow{a} T) \in \mathbb{C}\downarrow T}$.

This definition is closely related to the *pullback-retyping* used in [4]. In fact, as illustrated to the right, the action of a retyping operation $\langle \mathcal{F}, \varphi \rangle\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ is pulling back along φ_{id_T} followed by composition with $\mathcal{F}(\mathrm{id}_T)$, which is retyping along the span $T \leftarrow \mathcal{F}(T) \to T'$, according to [4].



The definition of grammar morphisms now is as follows.

Definition 14 (Typed grammar morphism). *Let* $G = \langle Q, s\colon S \to T \rangle$ *and* $G' = \langle Q', s'\colon S' \to T' \rangle$ *be typed grammars in* $\mathbb{C}$*. Then a* grammar morphism $\underline{\mathcal{F}}\colon G \to G'$ *is a retyping operation* $\underline{\mathcal{F}} = \langle \mathcal{F}, \varphi \rangle\colon \mathbb{C}{\downarrow}T \to \mathbb{C}{\downarrow}T'$ *such that*

(i) the start object is preserved, i.e. $\mathcal{F}(s) = s'$*, and*

[5] This means that $\varphi = \{\mathcal{F}(A) \text{–}\varphi_a\text{→} A\}_{a \in \mathbb{C}\downarrow T}$ is a family of arrows such that for each arrow $\psi\colon a \to b$ in $\mathbb{C}{\downarrow}T$, the span $A \leftarrow \varphi_a \text{–} \mathcal{F}(A) \text{–}|\mathcal{F}(\psi)|\text{→} \mathcal{F}(B)$ is a $\mathbb{C}$-pullback of $A \text{–}|\psi|\text{→} B \leftarrow \varphi_b \text{–} \mathcal{F}(B)$, yielding a pullback square $\begin{smallmatrix} B & \leftarrow & \mathcal{F}(B) \\ \uparrow & \lrcorner & \uparrow \\ A & \leftarrow & \mathcal{F}(A) \end{smallmatrix}$.

(ii) for any rule $q \in Q$, the image $\mathcal{F}(q) := \mathcal{F}(l_q) \leftarrow_{\mathcal{F}(\alpha_q)} \mathcal{F}(k_q) \rightarrow_{\mathcal{F}(\beta_q)} \mathcal{F}(r_q)$ *is either a rule in G', i.e. $\mathcal{F}(q) \in Q'$, or an identity rule, i.e. $\mathcal{F}(q) = \mathcal{F}(l_q) \overset{\mathrm{id}}{\leftarrow} \mathcal{F}(l_q) \overset{\mathrm{id}}{\rightarrow} \mathcal{F}(l_q)$.*

We will now prove the *Simulation Lemma:* it shows that each grammar morphism maps rewriting steps in the domain to corresponding ones in the codomain, which are either applications of rules in the target grammar or identity steps. Hence grammar morphisms preserve reachability.

Lemma 15 (Simulation Lemma). *Let $\underline{\mathcal{F}} = \langle \mathcal{F}, \varphi \rangle \colon G \to G'$ be a morphism between grammars $G = \langle Q, s \colon S \to T \rangle$ and $G' = \langle Q', s' \colon S' \to T' \rangle$. Then for any rewriting step $a \vDash_{\langle q,m \rangle \Rightarrow} b$ in G we have that $\mathcal{F}(a) \vDash_{\langle \mathcal{F}(q), \mathcal{F}(m) \rangle \Rightarrow} \mathcal{F}(b)$.*

Proof (Sketch). Suppose that $(A \text{–} a \rightarrow T) \vDash_{\langle q,m \rangle \Rightarrow} (B \text{–} b \rightarrow T)$, with match $m \colon l \rightarrowtail a$ for a rule $q = l \leftarrow_\alpha k \rightarrow_\beta r \in Q$. Using Proposition 6, there is a diagram of the form ${}^{l}_{a}\!\downarrow\!{}^{\leftarrow}_{\leftarrow}\!\lrcorner\!{}^{\rightarrow}_{\rightarrow}\!\downarrow\!{}^{r}_{b}$. As pushouts and pullbacks in $\mathbb{C}\downarrow T$ are constructed in $\mathbb{C}$, it is enough to consider the underlying $\mathbb{C}$-diagram ${}^{L}_{A}\!\downarrow\!{}^{\leftarrow}_{\leftarrow}\!\lrcorner\!{}^{\rightarrow}_{\rightarrow}\!\downarrow\!{}^{R}_{B}$. Now the morphism $\langle \mathcal{F}, \varphi \rangle$ provides not only arrows $\varphi_a \colon \mathcal{F}(A) \to A$ and $\varphi_b \colon \mathcal{F}(B) \to B$ into the "tips" of the two squares, but actually a pair of "fitting" pullback cubes over ${}^{L}_{A}\!\downarrow\!{}^{\leftarrow}_{\leftarrow}\!\lrcorner\!{}^{\rightarrow}_{\rightarrow}\!\downarrow\!{}^{R}_{B}$. The top face of the resulting double cube is ${}^{\mathcal{F}(L)}_{\mathcal{F}(A)}\!\downarrow\!{}^{\leftarrow}_{\leftarrow}\!\lrcorner\!{}^{\rightarrow}_{\rightarrow}\!\downarrow\!{}^{\mathcal{F}(R)}_{\mathcal{F}(B)}$, which by Proposition 6 is a Par($\mathbb{C}$) pushout square because RPCs and pushouts of pairs of monomorphisms are stable under pullback.

4 Unfolding grammars into occurrence grammars

Every typed grammar can be *unfolded* by recording all possible sequences of rewriting steps originating from the start object. In analogy to the constructions proposed for Petri nets and graph grammars, the structure that we will obtain is a *(non-deterministic) occurrence grammar*, which is a partial order representation of all possible computations. Finite initial parts of the (full) unfolding of the grammar – so-called *prefixes* – give a compact representation of the behaviour of the grammar up to a certain *causal depth*.

In this section we introduce the class of occurrence grammars and show that in such grammars reachable objects can be characterized statically, by means of suitably defined dependency relations (causality and asymmetric conflict) between rules. This allows to avoid "solving" reachability problems while constructing the truncations of the full unfolding, i.e. the algorithm presented at the end of this section builds the unfolding in a static manner.

4.1 Occurrence grammars

To properly define occurrence grammars, we need to recall from [2] the corresponding relations between rules of typed grammars, which can be described in words as follows. Given two rules q and q', then q *causes* q' if q produces something needed by q' to be activated, and q *can be disabled* by q' if q' destroys something on which q depends.

Definition 16 (Causality, conflict). *Let $G = \langle Q, s\colon S \to T\rangle$ be a typed grammar; then G is* mono-typed *if s is monic and, for each rule $q \in Q$ all three of l_q, k_q and r_q are monic, yielding subobjects $[l_q], [k_q]$ and $[r_q] \in \mathrm{Sub}(T)$. If G is mono-typed, a pair of rules $q, q' \in Q$ may be related in any of the following ways.*

$<$ **:** q directly causes q', *written* $q < q'$, *if* $r_q \sqcap l_{q'} \not\sqsubseteq k_q$
$\ll$ **:** q can be disabled by q', *written* $q \ll q'$, *if* $l_q \sqcap l_{q'} \not\sqsubseteq k_{q'}$

Further, the asymmetric conflict *relation is* $\nearrow := <^+ \cup (\ll \setminus \mathrm{id}_Q)$ *where* $<^+$ *is the transitive closure of* $<$ *and* $\ll \setminus \mathrm{id}_Q$ *the irreflexive version of* $\ll$*; moreover*

– *the* direct causes of q, *are given by* $\llcorner q \lrcorner = \{q' \in Q \mid q' < q\}$, *and*
– *the* (complete) causes of q, *are given by* $\lfloor q \rfloor = \{q' \in Q \mid q' <^* q\}$.

Any subobject $[a] \in \mathrm{Sub}(T)$ may be related to a rule $q \in Q$ in a similar way:

$<$ **:** q directly causes $[a]$, *written* $q < a$, *if* $r_q \sqcap a \not\sqsubseteq k_q$, *and*
$<\!\circ$ **:** $[a]$ *is* (partly) consumed by q', *written* $a <\!\circ\ q'$, *if* $a \sqcap l_{q'} \not\sqsubseteq k_{q'}$,

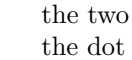
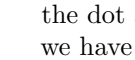
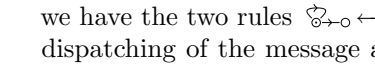
and we also have the following sets:

– *the* consumers of $[a]$, *are* $\ulcorner a \urcorner = \{q' \in Q \mid a <\!\circ\ q'\}$ *and*
– *the* (complete) causes of $[a]$, *are* $\lfloor a \rfloor = \{q' \in Q \mid \exists q \in Q.\, q' <^* q < a\}$.

Now we are ready to define *occurrence grammars*, which are a generalization of occurrence nets. We will see later that in an occurrence grammar with type object T, rule applications can be interpreted as consuming and producing subobjects of T (Proposition 19).

Definition 17 (Occurrence Grammar). *An* occurrence grammar *is a mono-typed grammar $O = \langle Q, s\colon S \rightarrowtail T\rangle$ with a countable set of rules Q such that*

1. *the type object is the union of all right hand sides, i.e.* $\mathrm{id}_T \cong s \sqcup \bigsqcup_{q \in Q} r_q$,
2. *the transitive-reflexive closure $<^*$ of causality $<$ is a partial order,*
3. *for each rule $q \in Q$, $\lfloor q \rfloor$ is finite, and $\nearrow|_{\lfloor q \rfloor} := \nearrow \cap (\lfloor q \rfloor \times \lfloor q \rfloor)$ is acyclic,*
4. *the start object has no causes, i.e.* $\lfloor s \rfloor = \varnothing$,
5. *there are no backward conflicts, i.e.* $r_q \sqcap r_{q'} \sqsubseteq k_q \sqcup k_{q'}$ *for all* $q \neq q' \in Q$,
6. *left-hand sides are properly produced, i.e.* $l_q \sqsubseteq s \sqcup \bigsqcup_{p' \in \llcorner q \lrcorner} r_{p'}$ *for all* $q \in Q$.

Example 18. Consider the following occurrence grammar which exactly captures the two rewriting steps of Examples 8 and 10. The type graph is  where the dot and the plus discern the different nodes; the start object is  , and we have the two rules  and $\circ \leftarrowtail \varnothing \rightarrowtail \varnothing$ corresponding to the dispatching of the message and the breakdown of the middle node. As it will become clear later, this grammar is part of the unfolding of the grammar in Examples 8 and 10.

Properties of occurrence grammars. In the theory of Petri nets, a characteristic property of occurrence nets is that they are safe, i.e. that every reachable marking is a set of places, rather than a proper multiset. The analogous result for occurrence grammars reads as follows.

Proposition 19 (Safety). *Let $O = \langle Q, S \succ s \rightarrow T \rangle$ be an occurrence grammar and let $s \vDash_O \Rightarrow^* a \in \mathbb{C} \downarrow T$ be a reachable object. Then a is monic.*

Hence reachable objects of occurrence grammars can be seen as subobjects of the type object. In the unfolding algorithm below, instead of considering reachable objects, we can concentrate on the statically characterized *concurrent subobjects*, as they are exactly the ones contained in reachable subobjects.

Definition 20 (Concurrent subobject). *Let $O = \langle Q, S \succ s \rightarrow T \rangle$ be an occurrence grammar. A subobject $[a] \in \mathrm{Sub}(T)$ is called a* concurrent subobject of O *if (i) $\lfloor a \rfloor$ is finite, (ii) $\lfloor a \rfloor \cap \ulcorner a \urcorner = \varnothing$, and (iii) $\nearrow|_{\lfloor a \rfloor}$ is acyclic.*

Intuitively, $[a]$ is concurrent when its set of causes is finite and conflict free (condition (i) and (iii), respectively) and there are no causal dependencies between subobjects of $[a]$ (condition (ii)).

Proposition 21 (Static coverability). *Let $O = \langle Q, S \succ s \rightarrow T \rangle$ be an occurrence grammar, and $[a] \in \mathrm{Sub}(T)$ be a subobject. Then $[a]$ is concurrent if and only if there is some reachable object b such that $a \sqsubseteq b$.*

4.2 The unfolding construction

The idea of the unfolding procedure for a given grammar G, is to construct a chain of growing occurrence grammars U_n. Each U_n represents all computations up to *causal depth* n where the depth of a concurrent computation is the length of a maximally parallel execution of the computation. Finally the full unfolding U_G will arise as the "union" of the chain $\{U_n \text{ “}\subseteq\text{” } U_{n+1}\}_{n \in \mathbb{N}}$. This is a concrete algorithmic description of the unfolding. As shown in the next section, the unfolding can be characterised in a succint and elegant way as the right adjoint functor to the inclusion of the category of occurrence grammars into the category of grammars.

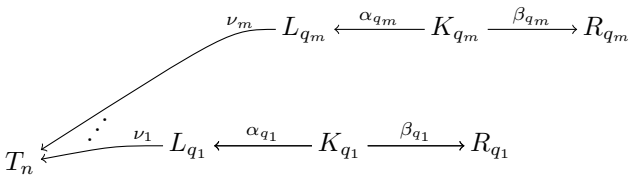
Definition 22 (Unfolding algorithm). Let $G = \langle Q, S - s \rightarrow T \rangle$ be a finite grammar. We will construct a chain $U_0 \text{ “}\subseteq\text{” } U_1 \text{ “}\subseteq\text{” } \ldots \text{ “}\subseteq\text{” } U_n \ldots$ of occurrence grammars $U_n = \langle Q_n, S - s_n \rightarrow T_n \rangle$ that come equipped with *folding morphisms* $\underline{\mathcal{F}}_n \colon U_n \rightarrow G$ mapping rule occurrences in each n-th unfolding U_n to the original grammar G; further each $\underline{\mathcal{F}}_n$ will be induced by a *folding arrow* $T_n - \lambda_n \rightarrow T$, i.e. $\underline{\mathcal{F}}_n = \sharp \lambda_n$ (see Definition 13).

Base case. The 0-th unfolding U_0 contains the start object of G and no rules, i.e. $U_0 = \langle \varnothing, S \succ \mathrm{id} \rightarrow S \rangle$. The folding arrow is $\lambda_0 \colon T_0 = S \rightarrow T$, which induces $\underline{\mathcal{F}}_0 = \sharp \lambda_0$.

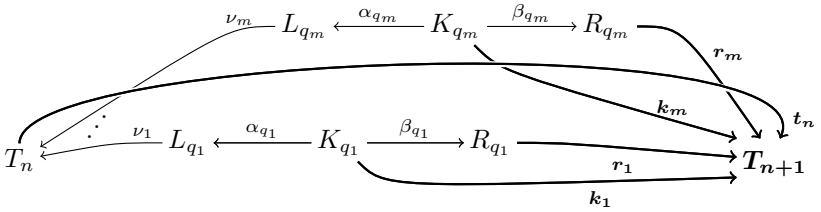
Induction step. Going from U_n to U_{n+1} consists in adding the next level of causal depth. The central operation of this step can be described as the *non-consuming* application of all rules with all possible (new) matches to T_n "in parallel" – here the non-consuming rule application of a rule $q = L \leftarrow \alpha \prec K \succ \beta \rightarrow R$ at a match $m \colon L \rightarrowtail T$ is the application of $q^+ := K \leftarrow \mathrm{id} \prec K \succ \beta \rightarrow R$ at $m \circ \alpha \colon K \rightarrowtail T$ (see Figure 1(c)).

A *new match* or a *new occurrence* of a rule $q \in Q$ in the n-th unfolding $\langle Q_n, S \xrightarrow{s_n} T_n \rangle$ via the folding $\mathcal{F}_n$ is a monomorphism $\nu \colon L_q \rightarrowtail T_n$ such that the corresponding subobject of $\mathrm{Sub}(T_n)$ is concurrent, and that satisfies $\lambda_n \circ \nu = l_q$; additionally, ν must be *new*, which means that ν is not an occurrence of q that is already present in Q_n, i.e. there is no rule $q' \in Q_n$ such that $\nu = l_{q'}$ and q is the image of q' w.r.t. $\mathcal{F}_n$.

Let $\{\nu_i \colon L_{q_i} \rightarrowtail T_n \mid i \in I_n\}$ be the set of all new matches where the index set $I_n = \{1, \ldots, m\}$ is finite as G is finite. Now consider the diagram below, consisting of the matches ν_i and the rule morphisms α_{q_i}, β_{q_i} for $i \in I_n$.



Take the colimit of the diagram above in $\mathbb{C}$, by a stepwise computation of pushouts of monos, obtaining the morphisms t_n, k_i, r_i into T_{n+1} for $i \in I_n$. Furthermore since every object in the diagram above is typed over T, we obtain the folding arrow $\lambda_{n+1} \colon T_{n+1} \to T$ as a mediating arrow.



Now the new rule occurrences form the set $Q'_{n+1} := \{(t_n \circ \nu_i) \leftarrow_{\alpha_{q_i}} k_{q_i} \rightarrow_{\beta_{q_i}} r_{q_i} \mid i \in I_n\}$ and are at depth level $n+1$; further the complete set Q_{n+1} of rules of U_{n+1} is $Q_{n+1} = Q'_{n+1} \cup \sharp t_n(Q_n)$.

To complete the object part of the $(n{+}1)$-th unfolding, we just need to define $U_{n+1} := \langle Q_{n+1}, S \rightarrowtail_{t_n \circ s_n} T_{n+1} \rangle$. Further the folding morphism $\mathcal{F}_{n+1} \colon U_{n+1} \to G$ is given by $\mathcal{F}_{n+1} := \sharp \lambda_{n+1}$, which is induced by the folding arrow $T_{n+1} \xrightarrow{\lambda_{n+1}} T$.

Summarizing, we have inductively defined an ω-chain of occurrence grammars $U_0 \text{“}\subseteq\text{”} U_1 \text{“}\subseteq\text{”} \cdots \text{“}\subseteq\text{”} U_n \ldots_\infty$, where each U_n has components $\langle Q_n, s_n \colon S \rightarrowtail T_n \rangle$, folding morphisms $\sharp \lambda_n \colon U_n \to G$, and “inclusion” morphisms $\sharp t_n \colon U_n \rightarrowtail U_{n+1}$.

Example 23. We sketch the unfolding of the grammar $\langle \{q'_1, q'_2\}, s \colon \text{⊙→○→⊛} \to \text{⊙○⊙} \rangle$ where the rules q'_1 and q'_2 are $\text{⊙→○} \leftarrowtail \text{○→○} \rightarrowtail \text{○→⊙}$ and $\circ \leftarrowtail \varnothing \rightarrowtail \varnothing$, respectively.

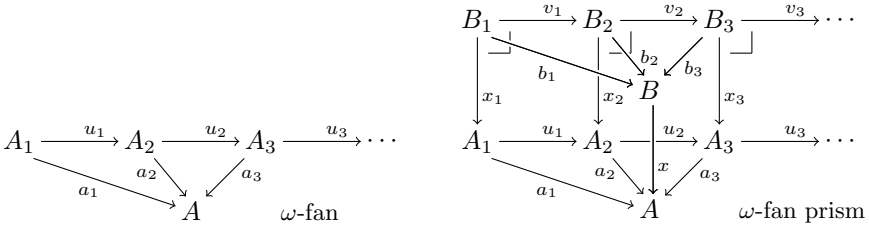
We start with the type graph $T_0 = \text{⊙→○→⊛}$. In the first unfolding step we find the single rule occurrence ⊙→○ of q'_1 and the three occurrences $\circledcirc$, $\circ$, and $\circledast$ of q'_2 and add the corresponding right-hand sides, yielding the type graph $T_1 = \text{⊙→⊙→⊛}$.

In the second unfolding there is only one occurrence of q_1, namely ⊙→⊛, and adding the right-hand side yields the type graph $T_2 = \text{⊙→⊙→⊙}$. Now as there are no further new matches, the latter unfolding is actually the full unfolding.

5 ω-adhesive categories and the coreflection result

In this section we propose ω-adhesive categories as a framework in which the unfolding construction is feasible and can be characterized as the right adjoint to the inclusion functor from the full sub-category of occurrence grammars into the category of all finite grammars. Note however that once we have the extra structure of ω-adhesive categories, we could also relax the condition on grammars from finite to countable.

As we mentioned at the beginning of Section 4.2, the unfolding U_G of a grammar G will be a single occurrence grammar that represents the complete chain of truncations generated by the algorithm of Definition 22. The colimits that we will use to construct U_G and to prove the coreflection result are *Van Kampen* (VK) *fans*: they are the ω-chain counterpart of Van Kampen squares, the latter being the central concept in the definition of adhesive categories in [14].



Definition 24 (ω-adhesive categories). *An* ω-fan *is an ω-chain diagram $\mathcal{A} = \{A_n \xrightarrow{u_n} A_{n+1}\}_{n\in\mathbb{N}}$ with a cocone $\alpha = \{A_n \xrightarrow{a_n} A\}_{n\in\mathbb{N}}$ (see the left one of the displayed diagrams); it is a* colimit ω-fan *if α is a colimit of $\mathcal{A}$, and it is a* Van Kampen fan *if in each ω-fan prism over it, as illustrated in the right one of the displayed diagrams, having pullback squares $\begin{smallmatrix} B_i & \to & B_{i+1} \\ \downarrow & \lrcorner & \downarrow \\ A_i & \to & A_{i+1} \end{smallmatrix}$ as back faces, the top face is a colimit ω-fan if and only if all lateral trapezia $\begin{smallmatrix} B_i & \to & B \\ & \searrow & \\ & A_i & \to & A \end{smallmatrix}$ are pullbacks.*

Now a category is ω-adhesive *if it is adhesive, and moreover*

- *it has colimits of monic ω-chains $\{A_n \rightarrowtail A_{n+1}\}_{n\in\mathbb{N}}$, and*
- *colimits of monic ω-chains give rise to Van Kampen fans.*

From now on, we assume $\mathbb{C}$ to be ω-adhesive. To ensure soundness of the full unfolding construction in Definition 26, we need the following lemma, which can be shown in analogy to Lemma 2.3 of [14].

Lemma 25 (Monic VK-fans). *Let $\mathbb{C}$ be any category, let $\{A_n \rightarrowtail A_{n+1}\}_{n\in\mathbb{N}}$ be a monic ω-chain paired with a cocone $\{A_n \xrightarrow{a_n} A\}_{n\in\mathbb{N}}$ such that they together form a Van Kampen fan. Then each $a_n \colon A_n \rightarrowtail A$ is monic.*

Definition 26 (Full unfolding). *Let $G = \langle Q, s \colon S \to T\rangle$ be a finite grammar, and let $\{U_n \rightarrowtail U_{n+1}\}_{n\in\mathbb{N}}$ be the chain constructed as in Definition 22, where $U_n = \langle Q_n, s_n \colon S \rightarrowtail T_n\rangle$ and $t_n \colon T_n \rightarrowtail T_{n+1}$ for each $n \in \mathbb{N}$. To define the* full unfolding *U_G, let $\iota = \{i_n \colon T_n \rightarrowtail T^U\}_{n\in\mathbb{N}}$ be the colimit of the ω-chain diagram $\mathcal{T} = \{T_n \rightarrowtail T_{n+1}\}_{n\in\mathbb{N}}$, and put $U_G := \big\langle \bigcup_{n\in\mathbb{N}} \sharp i_n(Q_n), i_0 \colon S \rightarrowtail T^U \big\rangle$.*

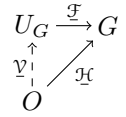
Finally, to define the folding morphism $\underline{\mathcal{F}}\colon U_G \to G$, *let the* $\lambda_n\colon T_n \to T$ *be as in Definition 22. By the universal property of the colimit* ι, *there is a unique folding arrow* $\lambda\colon T^U \to T$ *satisfying* $\lambda \circ i_n = \lambda_n$ *for all* $n \in \mathbb{N}$; *now put* $\underline{\mathcal{F}} := \sharp\lambda$.

Proposition 27 (Completeness of the unfolding). *Let* G *be a grammar and* $\sharp\lambda\colon U_G \to G$ *be the folding morphism from the full unfolding* U_G.

Then each derivation in G *has a unique counterpart in* U_G, *i.e. for each rewriting sequence* $s \vDash\langle q'_1,m'_1\rangle\Rightarrow a'_1 \cdots \vDash\langle q'_n,m'_n\rangle\Rightarrow a'_n$ *in* G, *there is a unique sequence* $s^U \vDash\langle q_1,m_1\rangle\Rightarrow a_1 \cdots \vDash\langle q_n,m_n\rangle\Rightarrow a_n$ *in the unfolding* U_G *such that* m'_i, q'_i, a'_i *are the images of* m_i, q_i, a_i *under the retyping with* λ.

This proposition is sufficient for many applications, but it does not rule out that U_G might contain superfluous information. The coreflection result ensures that the unfolding with the folding morphism $\underline{\mathcal{F}}\colon U_G \to G$ is the "minimal" or – more precisely – *universal* choice of an occurrence grammar O and a morphism $\underline{\mathcal{H}}\colon O \to G$ that is complete in the sense of Proposition 27.

Theorem 28 (Coreflection). *Let* $\underline{\mathcal{F}}\colon U_G \to G$ *be the folding morphism from the unfolding* U_G *of a finite grammar* G. *Then for each occurrence grammar* O *and morphism* $\underline{\mathcal{H}}\colon O \to G$ *there is a unique morphism* $\underline{\mathcal{V}}\colon O \to U_G$ *such that* $\underline{\mathcal{H}} = \underline{\mathcal{F}} \circ \underline{\mathcal{V}}$.



Proof (idea). Existence and uniqueness of some morphism $\underline{\mathcal{V}} = \langle \mathcal{V}, \zeta\rangle$ follow from two facts: first, the morphism $\underline{\mathcal{H}} = \langle \mathcal{H}, \vartheta\rangle$ determines ζ and $|\_|_{T^U} \circ \mathcal{V}$, and in fact the only information missing is the value $\mathcal{V}(\mathrm{id}_{T'})$ – here $O = \langle Q', s'\colon S' \to T'\rangle$; second, the type object T' is the "tip" of a VK fan, the diagram of which is determined by the start object and all the right hand sides. Pulling back this fan along $\vartheta_{\mathrm{id}_{T'}}$ yields again a colimit fan, and $\mathcal{V}(\mathrm{id}_{T'})$ arises as a uniquely determined mediating morphism. □

This theorem directly implies that the unfolding construction extends to a functor from the category of finite (or even countable) grammars to that of occurrence grammars, which in turn means that the category of occurrence grammars is a coreflective subcategory of the category of finite grammars.

As for examples and counter examples of ω-adhesive categories: the category $\mathbb{S}$ of *finite* sets – the "primordial" elementary topos – is *not* ω-adhesive; as a fact, any elementary topos is ω-adhesive if and only if it has countable sums. Hence the category of sets is ω-adhesive, and more generally any Grothendieck topos is. Further examples arise via the following constructions.

Proposition 29 (Closure of ω-adhesivity). *Let* $\mathbb{C}$ *and* $\mathbb{D}$ *be* ω*-adhesive categories. Then the following categories are again* ω*-adhesive:*

- *the product category* $\mathbb{C} \times \mathbb{D}$;
- *the slice category* $\mathbb{C}{\downarrow}T$ *for any* $T \in \mathbb{C}$;
- *the co-slice category* $I{\downarrow}\mathbb{C}$ *for any* $I \in \mathbb{C}$;
- *the functor category* $[\mathbb{X}, \mathbb{C}]$ *for any category* $\mathbb{X}$;
- *the Artin-Wraith glueing* $\mathbb{C}{\downarrow}\mathcal{F}$, *i.e. the comma category* $\mathbb{C}{\downarrow}\mathcal{F}$ *for any functor* $\mathcal{F}\colon \mathbb{D} \to \mathbb{C}$ *that preserves pullbacks.*

Proof (Sketch). In each case pullbacks and the relevant colimits are constructed componentwise.

In addition we know that the slice construction preserves stability of pseudo-complementation. Note also that all examples of "graph categories" mentioned in the introduction (undirected and directed graphs, hypergraphs, graphs with scopes, graphs with second-order edges, etc.) are ω-adhesive and RPCs are stable under pullback.

6 Related work and conclusion

Our work is strongly related to earlier work on true concurrency in the setting of adhesive categories. For instance [14] shows parallel and sequential independence results for adhesive rewriting systems. As a next step, in [2] it has been shown how one can represent computations of a system as processes, i.e. as deterministic occurrence grammars. In the present paper we generalize this work to non-deterministic occurrence grammars (or branching processes) that record all events of a set of possible computations.

The central contribution is the generalization of the unfolding technique to the abstract setting of ω-adhesive categories and the theorem that the unfolding construction, "unravelling" a grammar into an occurrence grammar, can be characterized as a coreflection. As this result holds in any ω-adhesive category with some mild restrictions it applies to numerous application-relevant instances of graph-like structures, and hence it is unnecessary to prove it again and again.

Furthermore we have introduced a new notion of grammar morphisms where the retyping is given by a functor. This allows us to treat also non-semi-weighted grammars, i.e. grammars where the start graph or the right-hand sides might not be injectively typed. Otherwise technical complications arise because of the presence of "too much symmetry" in the structure which is being unfolded and hence the uniqueness of arrow $\underline{\nu}$ in Theorem 27 cannot be guaranteed. Another solution to the symmetry problem has been proposed in [11], for the case of Petri nets and with a different notion of morphism.

Our definition of grammar morphism is not the most general one that can be conceived: it was inspired by previous works on graph transformation [1] and on unfolding constructions characterized as coreflections [20]. It is not obvious to what extent the coreflection result of Section 5 could be extended to more general definitions of morphisms: this is an interesting topic for future work.

The unfolding represents all computations as well as all reachable objects of the original grammar in a single acyclic branching structure. Hence, as observed in [17, 3], it can serve as the basis for partial order verification techniques. For instance, we plan to generalize the notion of finite complete prefix to the abstract framework of the present paper. Another direction is to adapt the model-based diagnosis techniques of [6, 5]; the latter depend on the preservation of products of grammars by the unfolding functor, which is ensured by the coreflection result. In future work we will further investigate products in our category of grammar morphisms.

References

1. P. Baldan. *Modelling Concurrent Computations: from Contextual Petri Nets to Graph Grammars*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 2000.
2. P. Baldan, A. Corradini, T. Heindel, B. König, and P. Sobociński. Processes for adhesive rewriting systems. In L. Aceto and A. Ingólfsdóttir, editors, *FoSSaCS*, volume 3921 of *LNCS*, pages 202–216. Springer, 2006.
3. P. Baldan, A. Corradini, and B. König. A framework for the verification of infinite-state graph transformation systems. *Information and Computation*, 206:869–907, 2008.
4. P. Baldan, A. Corradini, U. Montanari, and L. Ribeiro. Unfolding Semantics of Graph Transformation. *Information and Computation*, 205:733–782, 2007.
5. Paolo Baldan, Thomas Chatain, Stefan Haar, and Barbara König. Unfolding-based diagnosis of systems with an evolving topology. In *Proc. of CONCUR '08*, volume 5201 of *LNCS*, pages 203–217. Springer, 2008.
6. A. Benveniste, E. Fabre, S.Haar, and C. Jard. Diagnosis of asynchronous discrete event systems, a net unfolding approach. *IEEE Transactions on Automatic Control*, 48(5):714–727, 2003.
7. G. Birkhoff. *Lattice Theory*. American Mathematical Society, 1967.
8. R. Cockett and X. Guo. Join restriction categories and the importance of being adhesive. Unpublished manuscript, slides from CT'07 presentation available at `http://pages.cpsc.ucalgary.ca/~robin/talks/jrCat.pdf`.
9. H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and Concurrency in High-Level Replacement Systems. *Mathematical Structures in Computer Science*, 1:361–404, 1991.
10. H. Ehrig, M. Pfender, and H.J. Schneider. Graph-grammars: an algebraic approach. In *Proc. of IEEE Conf. on Automata and Switching Theory*, pages 167–180, 1973.
11. J. Hayman and G. Winskel. The unfolding of general Petri nets. In *Proc. of FSTTCS '08*, number 08004 in Dagstuhl Seminar Proceedings, 2008.
12. T. Heindel and P. Sobociński. Van Kampen colimits as bicolimits in Span. In *Proc. of CALCO '09*, LNCS. Springer, 2009. to appear.
13. P.T. Johnstone. *Sketches of an Elephant*, volume 1. Oxford Science Publications, 2002.
14. S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(2):511–546, 2005.
15. S. Lack and P. Sobociński. Toposes are adhesive. In A. Corradini, H. Ehrig, U. Montanari, L. Ribeiro, and G. Rozenberg, editors, *ICGT*, volume 4178 of *LNCS*, pages 184–198. Springer, 2006.
16. M. Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109:181–224, 1993.
17. K.L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.
18. J. Meseguer, U. Montanari, and V. Sassone. On the semantics of Place/Transition Petri nets. *Mathematical Structures in Computer Science*, 7(4):359–397, 1997.
19. E. Robinson and G. Rosolini. Categories of partial maps. *Inf. Comput.*, 79(2):95–130, 1988.
20. Glynn Winskel. Event structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer, 1987.